# Autonomous MCP-Based Job Application System

## Project Overview and Goals

The Autonomous MCP-Based Job Application System aims to develop an advanced, intelligent, and fully autonomous multi-agent framework utilizing MCP (Model Context Protocol) and cutting-edge AI models (GPT and Claude APIs). The system will automate the processes of job discovery, job evaluation, and application submission across multiple platforms, requiring minimal human intervention.

Core goals include: - **Intelligent Job Discovery:** Automatically identify and extract relevant job listings from platforms like LinkedIn, Indeed, and others. - **Dynamic Profile Integration:** Automatically extract and semantically interpret user profile data from multiple sources, including LinkedIn, resumes, and GitHub. - **Automated Preference Handling:** Utilize GPT and Claude models for intelligent, real-time synonym matching and preference standardization, significantly reducing manual maintenance. - **Structured Data Storage:** Implement scalable and structured storage of user profiles and preferences using SQLite, ensuring ease of future upgrades. - **Visualized Agent Orchestration:** Provide clear, real-time visual insights into agent interactions and workflows through MCP-based orchestration. - **Multi-user Scalability:** Design the system from inception to effectively support multiple users, each with unique profiles and job preferences.

## System Architecture and Communication

The architecture is built around a Coordinator Agent, which employs MCP to facilitate structured communication among specialized agents. The architecture distinctly separates intelligent decision-making tasks from practical, tool-based operations, promoting efficiency and adaptability.

MCP ensures structured and consistent agent-tool communication: - Agents exchange JSON-formatted MCP messages clearly detailing task definitions, required inputs, expected outputs, and status updates. - Messages explicitly guide next actions and any dependencies between agents.

## Key Modules and Agent Responsibilities

### Coordinator Agent

Manages task assignment and workflow across specialized agents using structured MCP messages. Ensures robustness through efficient error handling and retry mechanisms.

### Profile Management Module

* **Profile Manager Agent:** Centralizes and integrates professional profile information.
  + **Resume Parser Tool:** Semantically extracts and interprets resume content using PyMuPDF and GPT embeddings.
  + **LinkedIn Profile Scraper Tool:** Extracts detailed user profiles via Playwright, further enriched with GPT embeddings.
  + **GitHub Profile Tool:** Gathers relevant coding projects and experience via the GitHub API.

### Job Search Module

* **Job Search Agent:** Coordinates the automated retrieval and preliminary filtering of job listings.
  + **LinkedIn Job Scraper Tool:** Efficiently collects job data from LinkedIn.
  + **Indeed Job Scraper Tool:** Captures job data from Indeed.
  + Tools produce structured, consistent job data ready for further evaluation.

### Job Evaluation Module

* **Job Evaluator Agent:** Uses GPT/Claude APIs to perform detailed semantic evaluations, matching job listings with user profiles and preferences.
  + Outputs prioritized and clearly ranked job opportunities for streamlined processing.

### Preference Management Module

* **Preference Matcher Tool:** Dynamically matches scraped form labels and synonyms to canonical user preferences using GPT/Claude APIs, substantially reducing manual upkeep.
* Employs structured SQLite databases to manage user-specific preferences efficiently.

### Application Module

* **Application Agent:** Automates the creation and submission of tailored job applications.
  + **Company Info Tool:** Compiles comprehensive information about target companies.
  + **Document Customizer Tool:** Generates tailored resumes and cover letters using GPT/Claude APIs.
  + **Form Submission Tool:** Automates the filling and submission of application forms via Playwright.

### Tracker Module

* **Tracker Agent:** Records and manages the status updates of job applications within structured SQLite databases.
* Enables clear visualization and real-time monitoring through integrated dashboards.

## Implementation Plan

1. **Immediate MVP:** Our initial MVP to check basic functionality with minimal job search. Develops LinkedIn job scraping and dynamic preference matching using GPT APIs. Implements basic Coordinator Agent with structured MCP communication. Success criteria: Reliable LinkedIn scraping and accurate dynamic synonym matching.
2. **Enhanced Profile Integration:** Responsible for creating an updated representation of user’s employee information profile that automates embedding-based profile extraction from resumes and LinkedIn profiles. Establishes a unified profile management system. Success criteria: Effective automated extraction of professional profiles.
3. **Semantic Job Evaluation:** Implements robust GPT-driven semantic matching and job evaluation. Creates a clear, prioritized ranking system for job listings. Success criteria: Consistent, accurate semantic matching of job listings.
4. **Automated Application Generation:** Automates the creation and submission of customized documents and forms utilizing GPT and Playwright. Success criteria: Reliable and efficient automated application submissions.
5. **Multi-user Scalability:** Develops and implements a robust database schema supporting multiple users. Ensures clear, structured management of multi-user preferences and profiles. Success criteria: Scalable database management clearly supporting multiple users.
6. **Real-time Frontend Dashboard:** Deploys a Streamlit-based frontend dashboard providing real-time visibility of agent activities and application statuses. Success criteria: Clear, intuitive, and effective real-time dashboard visualization.

## Technical Specifications and Risk Management

### Technology Stack

* **Programming Language:** Python 3.11
* **Agent Orchestration:** LangChain, CrewAI (with optional LangGraph visualization)
* **Web Automation:** Playwright
* **Semantic Understanding:** GPT-4 and Claude APIs
* **Structured Data Storage:** SQLite (with planned upgrade path to PostgreSQL)
* **Frontend Visualization (Future):** Streamlit

### Decision-Making Rationale

* **MCP Integration:** Ensures a clear, structured, and future-proof communication framework among agents.
* **GPT and Claude APIs:** Provide advanced, reliable semantic understanding essential for dynamic synonym matching and comprehensive job evaluation.
* **SQLite Database:** Offers simplicity and sufficient initial scalability, with an easy transition path to more powerful databases if required.

### Risks and Mitigation Strategies

* **Cost Management:** Actively monitor and cache common GPT API calls to control costs.
* **Token Limits:** Optimize GPT prompts and implement caching to reduce token consumption.
* **Dynamic Web Interfaces:** Maintain highly modular scraper tools to swiftly adapt to website updates.
* **Data Privacy:** Implement secure credential management and encryption protocols for all user data.